Programming Assignment-5

Take Home Series ~ CSI 3150

Develop a used car sales interface using the technologies HTML, CSS , and JS.

1. On initial page load, the interface will show all the used cars available in the dealership (data provided as attachment for a few sample cars; you can add more to it if you like, it is not mandatory).
2. The layout and design of the interface is for you to decide but the css should be professional and presentable. Hint: if you want to display the car listings itself as it is generally shown in real car dealership websites, you may want to use the product card display layout (https://www.w3schools.com/howto/howto_css_product_card.asp) and refer to any car dealership websites for inspiration for CSS.
3. The interface must also support **all** the filter functionalities (mentioned below; to be implemented using JS) which will allow users to specify their desired values for the following filter fields (what html elements you use to implement them is up to you)
    a. Min. car year – Max. car year (e.g. 2005 -- 2012)
    b. Select one or more options from the available (car) make at the dealership (for e.g. Toyota, or Ford, or Lexus, etc.)
    c. Choose the desired max threshold value of car mileage (e.g. under 15000 or under 30000 miles etc.)
    d. Min. car price – Max. car price (e.g. 5,000 – 25,000) [or just specify the max price]
    e. Select one or more options from the available (car) colors (e.g. silver, white, black, etc)
4. Once the user specifies their desired filter option(s), the interface must show only the car listings that matches the filtering criteria. If no cars from the dataset matches the filter criteria set by the user, the interface must state the same to user and ask them to try again. In such a scenario, no car listings should be shown on the interface.
5. Hint 2: In order to "use" the data structure of the data file in your main JS file, simply link both the JS files in your HTML file using the script:src tag within the head tag of your HTML file. This will enable you bring in the data from the data file into an array of objects called usedCars inside your primary JS file where you can write the rest of the logic.
6. Exemption: If you want to statically create the html structure by manually typing in the data from the data file into your html, you may do so for the sake of this assignment, but try dynamic generation using JS (as demoed in class lectures) as static generation is not a viable solution for large datasets. However, if you do static generation, you will not be penalized.
7. Point distribution:
    a. Interface (html and css) look clean and professional [40 points]
    b. JS filter functionality for all fields are working as intended [60 points]

Happy Coding.

Submission Instructions [same as previous programming assignment]:

You must make use of Git and Github. Your final submission should be a PDF document which must list 2 links –

- Link to your github repository. Your public repository name must have the following naming convention - yourname_hw5TH_csi3150_f2024. And,
- Online link to access your web site. You can use "Github Pages" to host your website free (Further instructions will be given in class. For additional reference ahead of time you may refer to my YouTube video https://youtu.be/_kPN6tiEogw and syntax for git and github given in HTML 5 wrap up slides available in Moodle from slides 14 to 41) [if you choose to host your website with a different hosting service that is also acceptable. Provide the appropriate hosting link to your website.]

Directory Structure must be well organized and codebase must be well documented with usage of html semantic tags whenever possible.

Aspects that will cost you point penalty if not included (this is NOT an exhaustive list):

1. You must make use of sematic HTML tags as much as you can. Absence or minimal use of semantic tags (less than two or three per page) will cost you point penalty.
2. Not using Github and Github pages to submit your assignments.
3. Not following the directory structure as instructed
4. Code not documented as and when needed.
5. Syntactical or accessibility issues present in your final product.
6. Not following the general expected content outline as given in the instructions.